available or may not be supported by currently installed protocols or by rather hardware, additional users can change the bandwidth demand of a process.

Other house-keeping matters may be accounted for by the following code:

```
_BlueDotOpen(
    Handle          gh,
    ComponentInstance       self
){
    GlobalsPtr
    UniversalBitsPtr            ub;
    ProcessStuffPtr                     ps;
    Boolean         same;
    ProcessSerialNumber                 psn;
    GlobalWorld     saved:
    OSErr           err;
    ub = (UniversalBitsPtr) GetComponentRefcon((Component) self)
    if(!ub){
        if(!(ub = (UniversalBitsPtr)
NewPtrSysClear(sizeof(UniversalBits))))
                return(MemError()),
        saved = GetCurrentGlobalWorld();
        err = InitCodeResource();
        if(!err)
                err = InitLibraryManager(0, kSystemZone,
kNormalMemory);
        if(!err)
                if(!IsFunctionSetLoaded((TFunctionSetID)
kBlueDotFunctions)){
                        err = kASLMCodeNotLoadedErr;
                        CleanupLibraryManager();
        }
        SetCurrentGloblaWorld(saved);
        if(err)
                ub->hacker = true;
        SetComponentRefcon((Component) self, (long) ub);
    }
    GetCurrentProcess(&psn);
    for(ps = (ProcessStuffPtr) ub->processes.qHead; ps; ps =
    (ProcessStuffPtr)
ps->qLink)
        if(SameProcess(&psn, &ps->psn, &same) == noErr && same)
                break;
    if(!ps){
        if(!(ps = (ProcessStuffPtr)
        NewPtrSysClear(sizeof(ProcessStuff))))
                return(MemError());
        ps->psn = psn;
        ps->universals = ub;
        Enqueue((QElemPtr) ps, &ub->processes;
    }
    if(!(gh = NewHandleClear(sizeof(Globals))))
        return(MemError());
    HLockHi (gh);
    gp = *(GlobalsHandle) gh;
    SetComponentInstanceStorage(self, gh);
    gp->self = self;
    gp->process = ps;
    Enqueue((QElemPtr) gh, &ps->instances);
    UpdateValues(ub);
    return(noErr)
}
    pascal ComponentResult
_BlueDotClose(
    GlobalsHandle                       gh,
    ComponentInstance       self
){
    GlobalsPtr      gp;
    UniversalBitsPtr            ub;
    ProcessStuffPtr                     ps;
    if(!gh)
        return(noErr) ;
    gp = *gh;
    ps = gp->process;
    Dequeue((QElemPtr) gp, &ps->instances);
    DisposeHandle((Handle) gh),
    ub = ps->universals;
    UpdateValues(ub),
```

-continued

```
    if(ps->instances.qHead)
        return(noErr) ;
    Dequeue(QElemptr) sp, &ub->processes);
    DisposePtr((ptr) ps);
    UpdateValues(ub);
    if(ub->processes.qHead)
        return(noErr),
    if(!ub->hacker) {
        EnterCodeResource() ;
        CleanupLibrary Manager() :
        LeaveCodeResource() ,
}
    SetComponentRefcon((Component) self, 0) ;
    DisposePtr((Ptr) ub),
    return(noErr) ;
}
```

The code set forth above in the description of one embodiment of the present invention can be stored in a main memory, a read only memory, or a mass storage device, or in other external storage devices such as magnetic discs or other magnetic media. FIGS. **10** shows such an embodiment of a main memory array containing a set of program instruction that, when executed by a processor of a computer system, performs steps according to one embodiment of the present invention. The steps stored in the memory array corresponding to FIG. **10** include steps to be performed at the client (non-administrator) node. The steps stored in the memory array corresponding to FIG. **11**, however, include steps to be performed at the administrator node. It will be apparent that other means for storing programs are available, and that some systems provide several different sources of stored programs to the same processor. For example, application-level programs may be stored in main-memory or on a magnetic disc, while lower layer programs may be stored in a special cache or in ROM.

We claim:

1. In a computer network comprising nodes, a method of administering sending of teleconference data over the network comprising:

determining an allocated bandwidth corresponding to the sending;

communicating the allocated bandwidth to the nodes;

inhibiting use of bandwidth by any of the nodes in excess of the allocated bandwidth;

monitoring at least one nodal happiness factor;

adjusting the allocated bandwidth in response to the at least one nodal happiness factor;

dynamically measuring bandwidth use of program elements at a node; and

assigning bandwidth among program elements, such that the total of assigned bandwidth is not greater than said allocated bandwidth.

2. The method of claim **1**, further comprising:

determining for each program element at each node a desired bandwidth, the desired bandwidth being a total minimum bandwidth at which all program elements have sufficient bandwidth to operate at maximum speed; and

determining for each program element a happiness factor, the happiness factor being proportional to the assigned bandwidth and inversely proportional to the desired bandwidth.

3. In a computer network comprising nodes, a system configured to administer at least one teleconference over the computer network, the system comprising:

a means for determining an allocated bandwidth corresponding to the at least one teleconference;

a means for communicating the allocated bandwidth to the nodes; and

a means for inhibiting use of bandwidth by any of the nodes in excess of the allocated bandwidth;

a means for monitoring at least one nodal happiness factor;

a means for adjusting the allocated bandwidth in response to the at least one nodal happiness factor;

a means for dynamically measuring bandwidth use of program elements at a node; and

a means for assigning bandwidth among program elements, such that the total of assigned bandwidth is not greater than the allocated bandwidth.

4. The system of claim 3, further comprising:

a means for determining a desired bandwidth for each program element at each node, the desired bandwidth being a total minimum bandwidth at which all program elements have sufficient bandwidth to operate at maximum speed; and

a means for determining a happiness factor for each program element, the happiness factor being proportional to the assigned bandwidth and inversely proportional to the desired bandwidth.

* * * * *